Applying Functional TSP to a Maintenance Project

Ellen George and Dr. Steve Janiszewski
PS&J Software Six Sigma

Personal Software Process^SM (PSP^SM) is taught in the context of new code development and implementation. We frequently hear developers say that it will only work for new development and that their maintenance project is different. This article dispels that myth, demonstrating how PSP and Team Software Process^SM (TSP^SM) were successfully adapted to plan and manage a maintenance project. Readers will learn the key success factors in forming a functional project team and will learn how to emphasize and de-emphasize portions of the TSP process to better address the needs of a maintenance project.

The Personal Software Process^SM (PSP^SM) is a software development process originated by Watts Humphrey at the Software Engineering Institute (SEI) in the early to mid-1990s. By design, it is a high-maturity development process with all the features required to support a single developer. PSP is a measurement-driven process that includes planning, estimating, design, personal reviews, and testing. Its basic concepts can be extended for all software development life-cycle phases.

The Team Software Process^SM (TSP^SM) was developed in the late 1990s to add team-level practices to the PSP. By so doing, the TSP makes the PSP suitable for use in a commercial software development environment. TSP begins with a facilitated project launch process that generates a detailed project plan. The project plan includes a development strategy, a tailored development process, detailed size and effort estimates, earned value (EV) plans, a schedule, a quality management plan, and a risk management plan. The launch process is a team-building exercise designed to foster a sense of ownership and commitment and to produce a high-performance work team. The TSP continues to support project execution activities via a structured weekly project status meeting and all the management practices necessary to run a full-scale development project.

The PSP for Engineers course is taught in the context of new development where the students are asked to complete a series of 10 programming assignments. As a result, there is a perception that PSP and TSP will only work for new development projects. In this article, we will demonstrate how PSP and TSP were adapted to successfully plan and manage a maintenance project.

The Team

The Maintenance Team Dilemma

Maintenance projects deal with post-development support. In general, this can include operational support and implementation of new or enhanced features as well as defect investigations and fixes. Some maintenance projects just involve working off a backlog of problem reports and change requests. These projects are relatively easy to handle in a conventional TSP launch. Although it may be necessary to modify the estimation algorithm because there is not a good correlation between effort and the size of the change, there is a known list of tasks (the backlogged problem reports and change requests) that can provide the basic input to the TSP planning process.

“When preparing to launch a TSP team, it is a good practice to start by identifying a successful end state. What would management, the team, and the coach consider to be a successful launch and project?”

The situation gets more complicated when near real-time operational support is added to the mix of project tasks. The majority of project tasks may be unknown at the time of the launch because the operational anomalies have not even taken place yet. Nonetheless, the organization needs to allocate adequate resources, commit to scheduled completion dates for backlogged tasks, and manage the overall effort.

Organizations often have difficulty planning, staffing, and managing these sorts of maintenance projects due to the unpredictability of both the rate at which operational anomalies are reported and the effort required to respond to anomaly reports. There is frequently no correlation between the length of an anomaly investigation and the size of the resulting change. In fact, many anomaly reports do not even result in a change.

As a result, team members frequently feel that any attempt to estimate and plan the effort for a maintenance project is futile. And yet, these projects need to be staffed and managed.

This article shows how PSP and TSP were successfully used to plan and manage a maintenance project and how the team was able to build a useful estimating scheme in a project environment where it was common for developers to spend 75 percent of their time on unplanned event-driven anomaly investigations.

The Project

The project involved maintaining a large network-based financial services package originally implemented in C. The package was key to the company’s revenue stream and required real-time operational support as well as fixes and enhancements.

The maintenance team was composed of 12 PSP-trained software developers. Half the maintenance team was located at a site in the United States, while the remaining half was located at a site in Europe. There was a team lead at each location. The project manager worked in the United States. The TSP launch was held in the team’s U.S. office.

Is It a Team?

Typical work for this group of developers includes investigating reports of operational anomalies, upgrading legacy software to support new requirements, adding new features, investigating defects, and fixing defects. In the past, each developer considered himself or herself responsible for a different product. As it turned out, they were each responsible for a component of a single product. Each product component was built and released separately. There were no task or schedule dependencies between developers. Defects were generally localized to a component of the product. Up to this point, individuals only needed to be focused on

^SM TSP and PSP are service marks of Carnegie Mellon University.

their own component.

This contrasts strongly with the situation in new development where there are usually obvious dependencies in each person's work on tasks being done by other team members as the team works together to produce a product.

Watts Humphrey defines the *functional team* as a team with the following:

> … has a functional, rather than a product, mission. While all the members may do similar work, they do not develop a single product and their individual tasks are usually quite independent. … [An] example would be a maintenance group where each member handles the repair and enhancement of a product. While several of the members might occasionally work on elements to be integrated into a common release, they would usually work alone. [1]

The developers on this project fit the definition of a functional team perfectly. Consequently, the decision was made to launch the team using the SEI's variant of the TSP launch process for functional teams, TSPf [1].

Preparation

When preparing to launch a TSP team, it is a good practice to start by identifying a successful *end state*. What would management, the team, and the coach consider to be a successful launch and project? Answering these questions requires having well-defined launch and project goals and a strategy for attaining those goals.

Questions

What were the project goals? Why was this group of individual developers being launched as a team? What did the organization or the project manager think they would be able to accomplish as a team that they would not be able to accomplish as individuals?

We drilled down on these questions in a series of launch planning meetings with the project manager. Ultimately, the project manager was able to clearly articulate three very specific objectives that provided a compelling reason for these individuals to work together as a team:

- Spread knowledge among the team and broaden experience to reduce areas of risk and increase efficiency.
- Improve real and perceived quality to reduce customer-generated interruptions.
- Increase ratio of planned tasks to reactive unplanned activities.

If they were able to accomplish these objectives, it would benefit not only the organization, but the team members as well.

Once we had a well-defined set of management objectives for the team, the next set of questions addressed the actual TSP launch. What were some ways that the team could plan for the *unplanned*, high priority interruptions that are characteristic of near real-time operational support? What might they use as a size metric? Was there a conceptual design? What does quality mean when they are modifying a small number of lines relative to the size of the base code?

The coach's goal is not to decide the answers to these questions for the team, but rather to consider some of the possible answers and to make sure that the team collected the right project data prior to the launch so they would be able to make decisions based on data during the launch.

Through this exercise of strategizing an approach for conducting the launch, it became apparent that there were two overriding themes: commonality and repeatability.

Preparing the Project Manager

Commonality was the theme that would help the individuals to gel as a team. If they found that they had enough in common with one another and that they could benefit by taking advantage of the synergy, then surely they would come together as a team.

It was the program manager's job to define the management goals that would help set the stage for the team to gel. These goals had to serve two functions: setting a long-term vision for the project and the organization while providing short-term targets to help the team focus and come together.

We decided that the best approach would be to identify a long-term vision to provide a frame of reference. To support the vision, a series of short-term goals were developed. The team would be given three to four short-term goals with the expectation that they would be able to achieve them within about a month. The first set of short-term goals was selected so that they would be achievable with relatively low risk.

Since maintenance work can be difficult to predict and plan, we decided that the team would have a mini re-launch every four to six weeks. Each of these re-launches would provide the program manager an opportunity to roll out the next set of goals on the path to achieving his overall vision.

Preparing the Team Members

A necessary ingredient to this team's success was to get the team members to believe that there was a considerable amount of *repeatability* in their work and that this repeatability would lead to an improved ability to estimate. We asked the team members to start gathering PSP data on their tasks several weeks prior to the launch in an attempt to find the repeatability in their daily or weekly activities.

We found that the developers had two distinct categories of tasks. The first category was high priority interruptions. These interrupts could occur at any time. When they did, all other work had to be put aside. The second category of task was background work, which consisted of the tasks that the developers worked on when they were not reacting to high priority interruptions.

It was obvious that the developers would not be able to anticipate what interruptions would occur. However, it seemed that there might be a pattern to the quantity of high priority interrupt effort that was spent within a period of time. If the developers could quantify the percentage of time each week that was spent handling the high priority interrupts, then they would be able to create a budget of task hours for this category of activity.

The developers were asked to flag each task they worked on as one that they either knew about at the beginning of the week or one that came in during the week as an interrupt. While the percentage by category differed by individual, we found that there was clearly repeatability for each individual from week to week. The developers were able to use the data they collected during launch meeting No. 4 to plan for time to be spent on planned tasks as well as to create a budget for time to be spent on unplanned tasks. The planned tasks would then be estimated and planned just like any other PSP task. Whenever an interrupt came in, the developer would estimate it and plan it. Time for the interrupt would be allocated from the budget for unplanned tasks.

The Launch

A TSP launch consists of a series of scripted meetings. We tailored the standard TSPf meeting scripts by employing some special techniques designed to contribute to the success of the team by reinforcing the themes of *commonality* and *repeatability*.

Tailoring – Management and Team Goals

For each goal, we asked the team why it was important to them and what the impact to the team would be if they missed it. They were being asked to justify the need for each goal. In doing so, they began to internalize the importance of the goals, converging on a common understanding of what the goals

Activity	Entry Criteria	Exit Criteria
Problem Investigation	• Anomaly report, defect report, or improvement request with assigned due date and priority.	• Root cause identified and documented. • Anomaly report closed or reassigned. • Additional defect reports or improvement requests opened as necessary. • Solution investigation assigned if needed.
Solution Investigation	• Defect report or improvement request with assigned due date and priority.	• Authorized solution.
Problem Fix	• Defect report or improvement request with assigned due date and priority. • Authorized solution (optional).	• Problem fixed. • Defect report or improvement request has test pending status.
Conformance Test	• Test request opened. • Test scripts are received from exchange.	• Test completed. • Filled scripts sent to exchange or test cancelled. • Test results published.
System Build	• System build request opened. • All defect reports and improvement requests are cross-referenced to the build request and have test past status. • All code changes in the build are traceable to a cross-referenced defect report or improvement request.	• Working system with required documentation.

Table 1: *Maintenance Process*

really meant. The team decided to break down their goals into prioritized tasks to be interleaved with their product tasks.

Tailoring – Conceptual Design

The team did not think of themselves as part of a bigger project so they were initially hesitant to spend time reviewing the conceptual design

We asked the design manager to project a system diagram on a screen and lead a discussion in which the *person most knowledgeable* with each component of the system briefed the others on the size of the component, interactions with other components of the system, and areas of risk.

This was the first time that the team members ever had the opportunity to see their own work on individual package components in an overall system context. The discussion became animated and the team members became *physically involved*[2]. In spite of their initial reticence, this was the meeting where individuals started to come together as a team.

Figure 1: *Anomaly Investigation Effort Histogram*

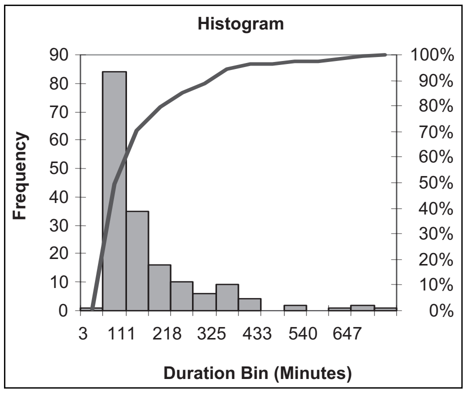

Table 2: *Anomaly Estimation Table*

	Duration (minutes)
Very Small	6
Small	18
Medium	59
Large	187
Very Large	596

Tailoring – Process Plan

The organizational maintenance process had been defined prior to the launch. So, the team focused its discussion on defining explicit entry criteria, exit criteria, and required approvals for moving from one process step to the next. The team was able to identify gaps in the defined process and to recommend modifications to address the gaps. The resulting maintenance process was relatively simple and is shown in Table 1.

The source of the problem could be a reported anomaly, a defect report, or a request of new functionality. Anomaly reports do not necessarily require product changes to be resolved. They could be operator errors, procedural issues, etc. The scope of the solution analysis depends on the size of the required change, and it could be omitted for a trivial change. Conformance testing verifies that the software will work in the exchange environment. It requires test scripts recorded from exchange feeds. Multiple changes are aggregated into a system build and released to production.

Tailoring – Top Level Plan

The team used the data they had collected over the past several weeks and analyzed the average number of task hours per week that they had each spent on *planned* versus *unplanned* tasks. Using this data, they were able to confidently plan for how many hours they could get on task each week, budgeting a percentage of those hours for unplanned or interrupt-driven tasks.

For many of the team members, approximately 75 percent of their task hours were being spent on interrupt-driven tasks. So, those team members who were achieving a total of 12 hours on task per week planned to spend three hours per week on background tasks and budgeted nine hours a week for interrupt-driven tasks.

Tailoring – Quality Plan

The team spent considerable time discussing what quality meant to their customer. A predominant concern was defects that were returned to the team by the customer after an initial fix. The team put metrics in place to explicitly track and manage the following:

- Quantity and frequency of returned *fixes*.
- Defect investigation time.
- Defect turnaround time.

Tailoring – Detailed Plan

One of the goals of this launch was to build a plan in which the team would close out all of their high priority tasks. The background tasks were sequenced so that the high-priority tasks would be completed first. Multiple EV plans were generated so that management would have visibility into the following:

- How long it would take to complete the high priority tasks given the level of interrupts that the team was experiencing.
- How much backlog work the team would complete in six weeks.
- How long it would take to work off the full backlog of tasks.

This was easily accomplished with the help of an automated scheduling tool.

Post-Mortem

The participants felt that "team synergy was improved," that they did a "good job of balancing work load," and that there was "exposure of everyone else's jobs" with "good participation and contribution from everyone." One participant summarized the experience: "This seems like the birthday of this team."

Results

Estimating

During the launch, the team was highly skeptical about their ability to estimate anomaly investigation tasks. However, they agreed to make their best estimates and then to develop an estimating algorithm from the post-launch data. Without historical data, the team estimating error was 41 percent. With the data gathered during the launch post-mortem, the team was able to reduce the estimating error to 23 percent.

Anomaly Investigation Effort

After the launch, the team collected data on the durations of anomaly investigation tasks for several months. A histogram[3] of the data, as shown in Figure 1, indicates that the duration of anomaly investigation tasks can be modeled as a random variable and that a skewed-distribution function, like a log normal, would provide a good basis for estimating the probable duration of an anomaly investigation task.

From the fitted distribution function, it is possible to determine what the expected duration of a very small, small, medium, large, and very large task would be, as shown in Table 2. The average length of an investigation task was 59 minutes, with approximately 70 percent of all investigations requiring between 18 and 187 minutes.

Estimating Algorithm

This result leads directly to a simple technique for estimating anomaly investigations that can be employed during a TSP launch. For the tasks that are part of the backlog for a special anomaly investigation, categorize each one as very small, small, medium, large, or very large and use the estimated time provided by the table. Prorate the total for all backlogged tasks to account for the unplanned investigations by using the historical percentage of time devoted to unplanned anomaly investigations. For this team, the percentage of time devoted to unplanned anomaly investigations was 75 percent to 85 percent of its available time.

Re-Estimating the Launch

During the launch, the team had estimated the effort of each identified investigation as small, medium, or large. Assessment of effort had been made by the task owner, based on familiarity with the functionality and amount of code that would need to be reviewed.

During post-mortem, the original small/medium/large estimate from the launch was used along with the calculated values of small, medium, and large to re-estimate the tasks. As shown in Table 3, the re-estimate based on the calculated size ranges reduced the estimation error by a factor of two to a total error of 23 percent.

	Actual Time	Est. Time (Launch)	Est. Time (S/M/L)
Minutes	2,549	4,301	3,313
% Error		- 41%	- 23%

Table 3: *Comparisons of Estimates*

Lessons Learned

The coach must prepare for every launch. The coach needs to understand the peculiarities of each team, anticipate potential trouble spots relative to planning, and have a strategy for how to facilitate the launch to engage the participants and to build a plan that supports the business objectives. Choose no more than three to four very specific and achievable goals to help unite the participants as a team. Use PSP 1.0 to collect data prior to the launch, and focus the team on actual data to help team members find the repeatability in their work and to make fact-based decisions during the launch.

Even though an individual task may be completely unpredictable, once the statistics that characterize a set of typical tasks are known, it is possible to use those statistics to make reasonably accurate estimates about the effort required to handle the normal workload associated with many *unpredictable* tasks. This allows a team to allocate the right number of resources to meets its commitments and bring a sense of control and predictability into an apparently chaotic project.◆

References

1. Humphrey, Watts. Coaching Development Teams. Addison-Wesley (in press to be published in 2005 or 2006).

Notes

1. TSPf is still in the prototype stage, but the SEI authorized piloting its use with this team.
2. Our experience has been that there comes a point in every successful launch where the team members get physically involved in the meetings. They sit up straighter in their chairs, leaning forward to hear better, or stand up and move chairs from the back of the room to the front so that they can see better or come forward to write on the white board.
3. The histogram shows the number of data samples falling into each of the duration bins indicated on the x-axis (bar chart) and the cumulative number of data samples with duration less than the x-axis value (curve). For example, there are 83 anomaly investigation tasks with duration between three and 111 minutes and approximately 65 percent of the data points have a duration less than 111 minutes.

About the Authors

Ellen George is a principal with PS&J Software Six Sigma and is active in training and consulting on the application of software process and project management. She has over 20 years of experience in software development, process improvement, and project management. George is a Software Engineering Institute authorized Personal Software Process^{SM} instructor and Team Software Process^{SM} launch coach. She has a Master of Science in computer science and a master's degree in technology management from the Stevens Institute of Technology.

PS&J Software Six Sigma
P.O. Box 463
Palisades Park, NJ 07650
Phone: (201) 358-8828
E-mail: ellengeorge@softwaresixsigma.com

Steve Janiszewski, Ph.D., consults on project management, Capability Maturity Model® Integration, software metrics, Personal Software Process^{SM}, Team Software Process^{SM}, and Six Sigma for software development for PS&J Software Six Sigma. He has 30 years of experience in all phases of software development, management, and process improvement. Prior to joining PS&J, Janiszewski was director of Honeywell's corporate Software Six Sigma organization, providing process assessments, training, and consulting to more than 100 locations around the world. He has a Master of Science and doctorate in theoretical physics from New York University.

PS&J Software Six Sigma
P.O. Box 463
Palisades Park, NJ 07650
Phone: (201) 947-0150
E-mail: stevejaniszewski@softwaresixsigma.com